Retrieval, Retrieval-Augmented Generation, and Summarization

CS 6120 Natural Language Processing

Northeastern University

Si Wu

Some slides are based on Jurafsky and Martin chapter 11

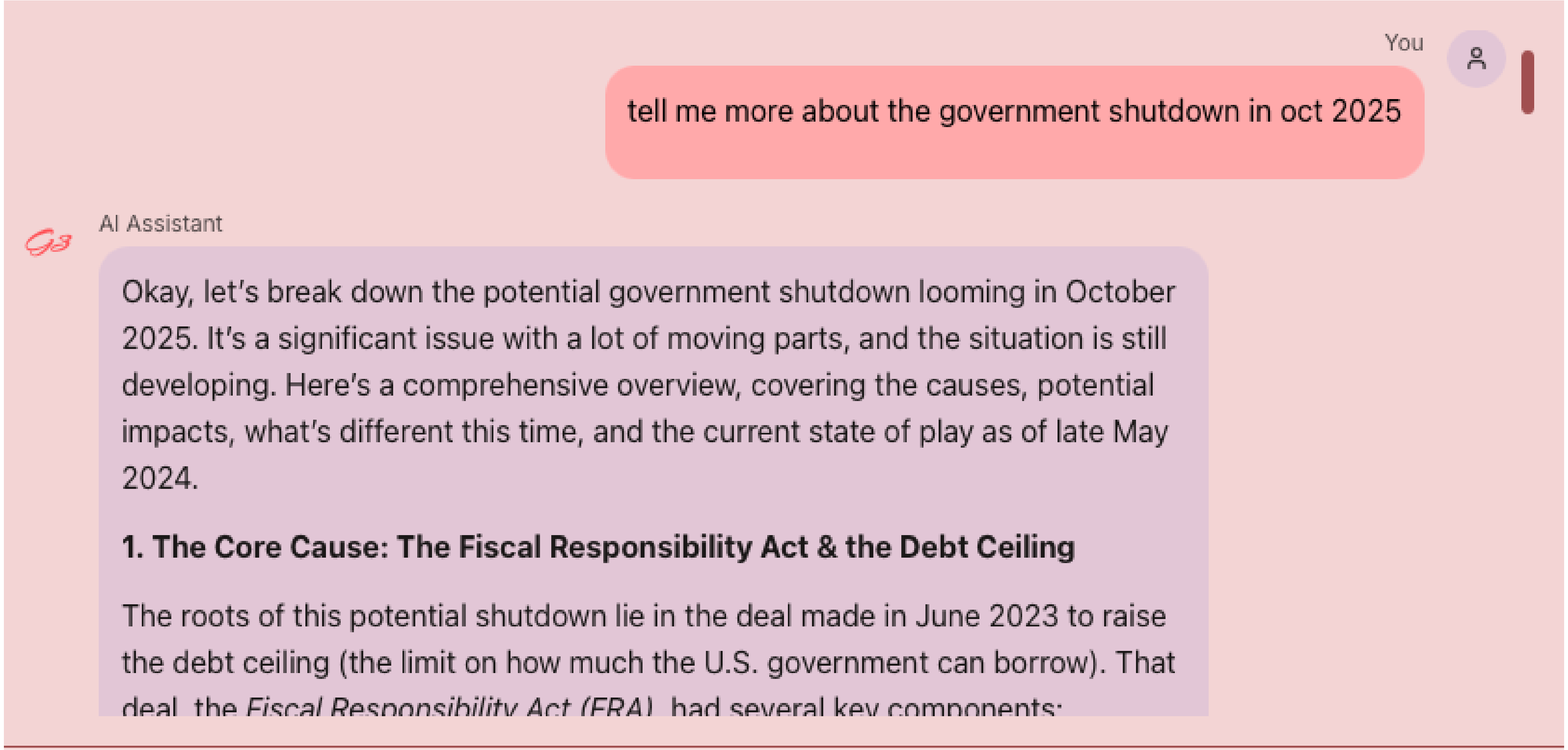
Logistics

- Data and experimental design is due this Friday
- Quiz policy change (both sessions):
  - Only 5 quizzes in total instead of 6
  - We will announce the next two quizzes at least the lecture before
  - we will drop the quiz with lowest grade at the end
    - Or keep all the quizzes, whichever give you the highest overall grade
  - This change should give you better grade overall

- Today:
  - Retrieval-augmented generation
  - Summarization

Retrieval-Augmented Generation

LLM knowledge is static



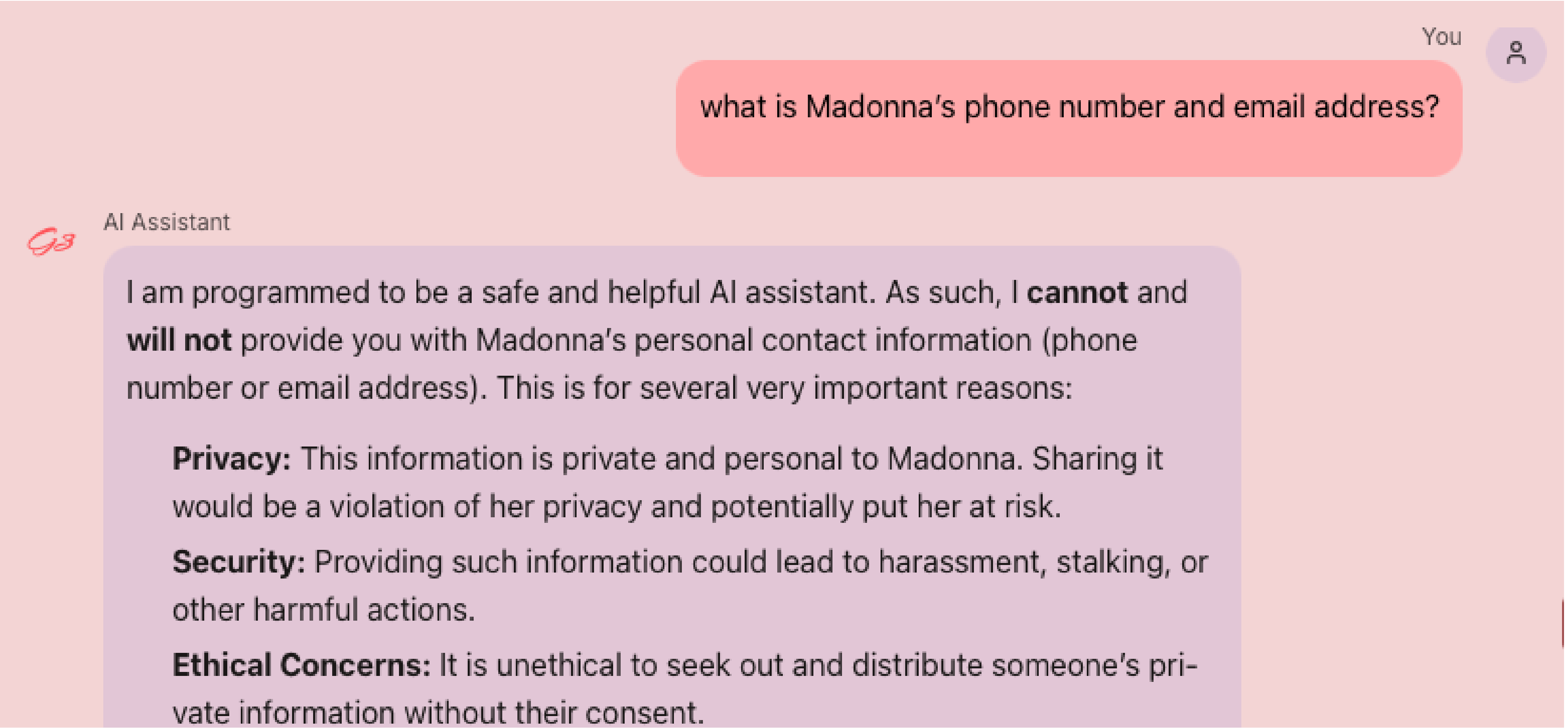
Examples of hallucination in summarization task

Article: ...the volcano was still spewing ash on Sunday, hampering rescue operations. More than a dozen people were killed when it erupted in 2014 ...rescue teams are still scouring the area, looking for more victims who may have been killed or badly burned ...
Summary: Rescue teams in Indonesia are searching for more than 20 people missing after the Mount Sinabung volcano erupted on Saturday, killing at least 11 people and injuring at least 20 others.

Article: The government and the doctors' union have agreed to continue negotiating until Wednesday. The talks, hosted by conciliation service Acas ...
Summary: Talks aimed at averting the imposition of a new junior doctors' contract in England have been extended for a second day.

Examples from Zhao et al. (2020)

Safety protocols in LLMs



Problems with LLM

- Knowledge is static:
  - Once we pretrained an LLM, it does not know things happened after that
- It’s not always feasible to tell if an LLM is hallucinating
  - Hallucination: a response that is not faithful or factual
- Sometimes an LLM may not allow us to ask certain questions about proprietary data, but we might want it for a good reason:
  - Healthcare: need medical records of a patient
  - Legal: legal discovery from proprietary documents

Retrieval-Augmented Generation (RAG)

- Solution to all the problems we just mentioned
- Give an LM **an external source of knowledge**
  - For example, proprietary texts like medical or legal records, personal emails, or corporate documents
- RAG uses Information Retrieval (IR) techniques to retrieve documents that are likely to have information that might help answer the question.
  - Then we use an LLM to generate the answer.
- Can be used for question answering (QA), summarization, etc.

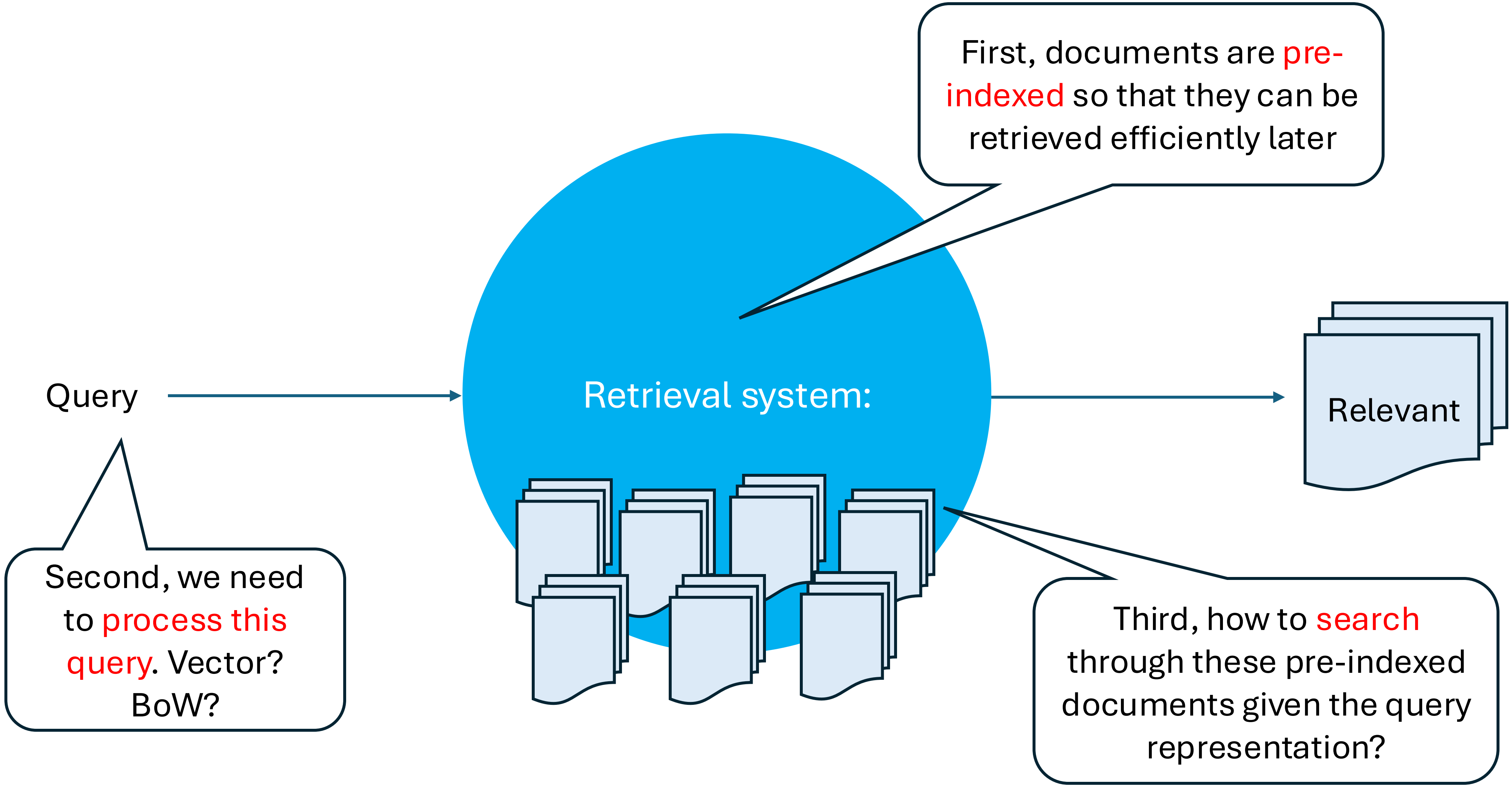
Retrieval-Augmented Generation (RAG)

- It helps ensure that the answer is **grounded in facts** from some curated dataset
  - The system can give the user the answer along with the relevant passage in the document
  - Help users have confidence in the accuracy of the answer, or help them spot when it is wrong.
- Today, in the age of LLMs, we generally have two separate systems: the retrieval system and the text generation system.
  - The retrieval system will handle a large collection of media, index them, and retrieve information efficiently given a query
  - How we integrate the two is an engineering choice.

Retrieval-Augmented Generation (RAG)

- Two main jobs of a retrieval system:
  - Index a collection of media
    - How should we index them?
      - Possible techniques: databases, hashing, etc.
  - Retrieve relevant media upon a query
    - How to retrieve efficiently?
    - Should we go through each instance one by one at every query?
    - Efficient retrieval methods reply on pre-computed indexes and similarity search techniques

A simplified illustration of an IR system

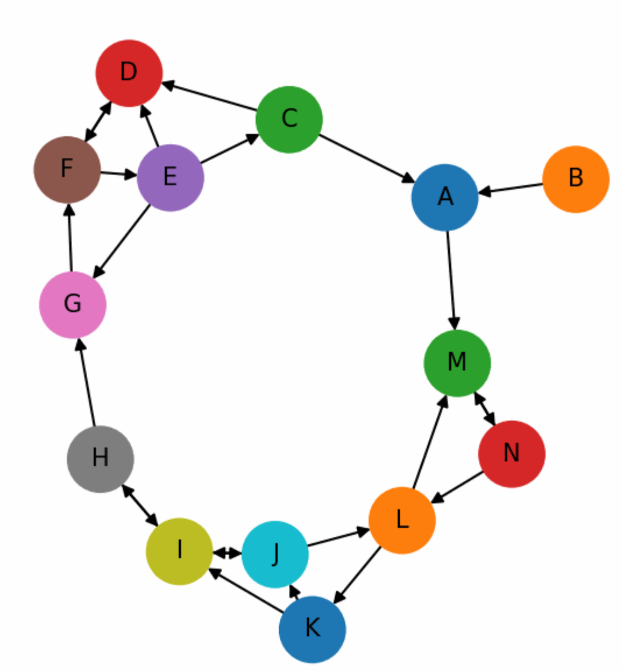


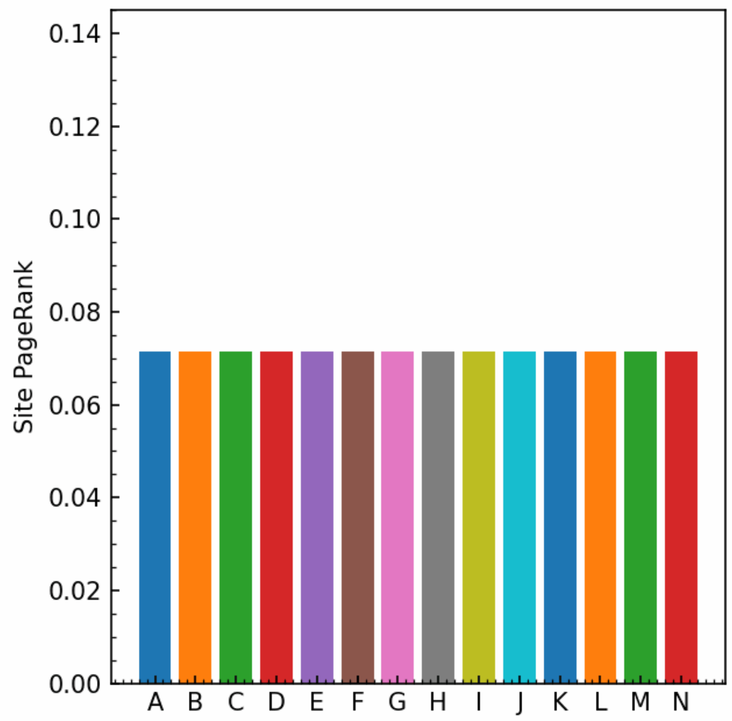
Information Retrieval

Just the basics

Even if you didn't take any NLP, Information retrieval, or even machine learning classes before, you probably took linear algebra in college and learned about **PageRank algorithm**...

That is information retrieval!





What's PageRank again?

- It's an algorithm that measures the **importance** of a webpage based on the number and quality of links pointing to it
- Each page distributes its rank equally among the pages it links to
- Each page's rank is influenced by the ranks of pages linking to it, weighted by how many links those pages have
- Repeat this process iteratively: modeling a random surfer who follows links until the rank values converge to stable probabilities

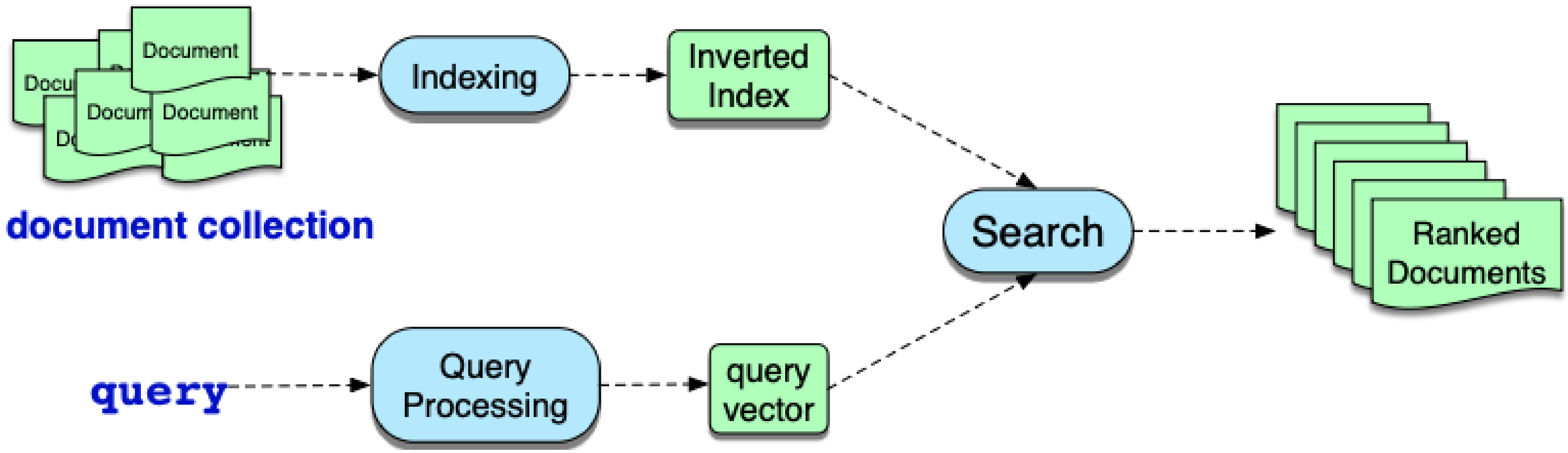
Larry **Page**

Basic IR concepts

- Information Retrieval (IR) is the name of the field encompassing the retrieval of all kinds of media based on user information needs
  - Image
  - Text/documents
  - Audio
- The resulting IR system is often called a **search engine**
- **Ad hoc retrieval:** a user poses a **query** to a retrieval system, and the system returns an ordered set of **documents** from some **collection**.
  - A **document**: whatever unit of text the system indexes and retrieves
    - web page, scientific paper, news article, or a paragraph

Basic IR concepts

- A **term**: a word in a collection. It could also include phrases
- A **query**: a user's information need expressed as a set of terms.



The query processing stage

- We have some query; how do we actually use it?
  - Split into words?
  - Vector representation?
- IR and NLP are closely related
- Just like in NLP, we started with n-gram, to now using embeddings
  - A query can be transformed into unigrams
  - Or we can use word or sentence embeddings with an **encoder**

Why are (dense) embeddings better?

- **Vocabulary mismatch problem**: we can’t find the exact word
  - Between query and document

- Instead of (sparse) word-count vectors, use (dense) embeddings
  - Any encoder models that can encode a text to an embedding: CNN, LSTM, BERT!

Indexing the documents

- How to index the documents? How do you store?
  - Simplest: exact word/term matching: “Romeo” → [doc1, doc2, doc3]
    - Examples: Tf-idf, BM25
    - Efficient, interpretable, but can’t capture semantics!
  - Hashing and other approximate indexing!
    - Good when you use high-dimensional vectors
    - Generally use similarity search
    - Very scalable
    - Not as good as newer approaches
  - Hybrid indexing: (combine the two above) lexical and semantic
    - Slower but better precision and recall
    - Examples: ColBERT
  - And many others: vector indexing, tree-based, quantization-based, etc.

Indexing the documents

- What is a document? What is your unit?
  - For a book chapter:
    - Do you embed the entire chapter as one embedding?
    - Or each paragraph?
    - Or each sentence?
    - Or have multiple embeddings for different hierarchy and use their weighted sum?

Retrieving the relevant pre-indexed documents upon a query

- Now that we have the query representation, and we have documented indexed, how do we get the “relevant” pre-indexed documents?
  - Use exact terms: tf-idf
  - Using **similarity**!

Without embeddings, n-gram approach: Term weighting and document scoring

- All documents are just bag of words
- **Term frequency** ($tf_{t,d}$): how frequent a term t appear in a specific document d.
  - Words that occur more often in a document are likely to be informative about the document's content
- **Document frequency** (df_t): how many documents a term t occurs in
  - Terms that occur in only a few documents are useful for those documents (maybe of a specific topic)
  - Terms that occur in all documents aren't really great discriminators
- Tf-idf:
  - idf_t: **inverse document frequency**

total number of documents

$$\mathrm{idf}_t = \log_{10} \frac{N}{\mathrm{df}_t}$$

Tf-idf

For a specific term t

$$\text{Tf-idf}(t,d) = TF(t,d) \;\; \times \;\; IDF(t)$$

How often does this word t appear **in this document d**

how rare or common this word is **in this collection of documents**

Example of tf-idf

Word	df	idf
Romeo	1	1.57
salad	2	1.27
Falstaff	4	0.967
forest	12	0.489
battle	21	0.246
wit	34	0.037
fool	36	0.012
good	37	0
sweet	37	0

Corpus: Shakespeare's plays

Romeo:
df =1 → only appear in 1 document

good, sweet:
df = 37, appeared in every play. Not very discriminative words

Document scoring

- Given a document d and a query q, we use their cosine similarity to measure their relevance to each other

$$\text{score}(q,d) = \cos(\mathbf{q},\mathbf{d}) = \frac{\mathbf{q}\cdot\mathbf{d}}{|\mathbf{q}||\mathbf{d}|}$$

Mathematically equivalent but another way to see it:

$$\text{score}(q,d) = \cos(\mathbf{q},\mathbf{d}) = \frac{\mathbf{q}}{|\mathbf{q}|}\cdot\frac{\mathbf{d}}{|\mathbf{d}|}$$

Normalize each to a unit vector first

Document scoring with tf-idf

$$\text{score}(q,d) = \cos(\mathbf{q},\mathbf{d}) = \frac{\mathbf{q}}{|\mathbf{q}|} \cdot \frac{\mathbf{d}}{|\mathbf{d}|}$$

Using tf-idf values for query and document, going through each term in the query:

$$\text{score}(q,d) = \sum_{t \in \mathbf{q}} \frac{\text{tf-idf}(t,q)}{\sqrt{\sum_{q_i \in q} \text{tf-idf}^2(q_i,q)}} \cdot \frac{\text{tf-idf}(t,d)}{\sqrt{\sum_{d_i \in d} \text{tf-idf}^2(d_i,d)}}$$

Example of score(q,d) with tf-idf

Query:	sweet love
Doc 1:	Sweet sweet nurse! Love?
Doc 2:	Sweet sorrow
Doc 3:	How sweet is love?
Doc 4:	Nurse!

Query

word	**cnt**	**tf**	**df**	**idf**	**tf-idf**	**n'lized** = tf-idf/$\lvert q \rvert$
sweet	1	1	3	0.125	0.125	0.383
nurse	0	0	2	0.301	0	0
love	1	1	2	0.301	0.301	0.924
how	0	0	1	0.602	0	0
sorrow	0	0	1	0.602	0	0
is	0	0	1	0.602	0	0

$|q| = \sqrt{.125^2 + .301^2} = .326$

Document 1

word	**cnt**	**tf**	**tf-idf**	**n'lized**	$\times$ **q**
sweet	2	1.301	0.163	0.357	**0.137**
nurse	1	1.000	0.301	0.661	0
love	1	1.000	0.301	0.661	**0.610**
how	0	0	0	0	0
sorrow	0	0	0	0	0
is	0	0	0	0	0

$|d_1| = \sqrt{.163^2 + .301^2 + .301^2} = .456$

Cosine: $\sum$ of column: **0.747**

Example of score(q,d) with tf-idf

Query:	sweet love
Doc 1:	Sweet sweet nurse! Love?
Doc 2:	Sweet sorrow
Doc 3:	How sweet is love?
Doc 4:	Nurse!

			Document 2		
word	**cnt**	**tf**	**tf-idf**	**n'lized**	$\times$**q**
sweet	1	1.000	0.125	0.203	**0.0779**
nurse	0	0	0	0	0
love	0	0	0	0	0
how	0	0	0	0	0
sorrow	1	1.000	0.602	0.979	0
is	0	0	0	0	0

$$|d_2| = \sqrt{.125^2 + .602^2} = .615$$

Cosine: $\sum$ of column: **0.0779**

			Query			
word	**cnt**	**tf**	**df**	**idf**	**tf-idf**	**n'lized** = tf-idf/$\lvert q \rvert$
sweet	1	1	3	0.125	0.125	0.383
nurse	0	0	2	0.301	0	0
love	1	1	2	0.301	0.301	0.924
how	0	0	1	0.602	0	0
sorrow	0	0	1	0.602	0	0
is	0	0	1	0.602	0	0

$$|q| = \sqrt{.125^2 + .301^2} = .326$$

BM25

- A more complex variant of simple tf-idf
- 2 parameters:
  - k: adjust the balance between tf and idf
  - b: control the importance of document length normalization

$$\sum_{t \in q} \overbrace{\log\left(\frac{N}{df_t}\right)}^{\text{IDF}} \overbrace{\frac{tf_{t,d}}{k\left(1-b+b\left(\frac{|d|}{|d_{\text{avg}}|}\right)\right)+tf_{t,d}}}^{\text{weighted tf}}$$

Average doc length in
the collection

Stop words

- Words like “the” “a” “an” “of” are of high frequency, and are not discriminative
- They should be removed for tf-idf
- A list of such words is called a **stop list**

Searching for documents with term t

- Given a large collection of documents, how to efficiently find documents that contain the term t?
- We use the data structure called **inverted index**

Term and num of documents that contain it

how {1}	→	3 [1]
is {1}	→	3 [1]
love {2}	→	1 [1] → 3 [1]
nurse {2}	→	1 [1] → 4 [1]
sorry {1}	→	2 [1]
sweet {3}	→	1 [2] → 2 [1] → 3 [1]

Document ID and Term frequency

BERT

- Tf-idf is terms a document into a bag-of-words representation
- Vocabulary mismatch is a big problem!

One solution to this

→Using contextual embeddings to encode terms or documents can capture meanings/semantics

Many different approaches to encode and score the similarity between document d and query q using BERT

Using a single BERT encoder

- Primer:
  - BERT is an encoder-only transformer model
  - Special token [CLS] → classify
  - Special token [SEP] → separate two sentences

[CLS] q_tokens [SEP] d_tokens [SEP]

[CLS] treated as the joint representation of both q and d

Using a single BERT encoder (cont.)

- The idea is you present both the query q and the document d to a single encoder
- Thus building a representation that is sensitive to the meaning of both q and d
- Use [CLS] output embedding as the joint representation of q and d
- Then a linear layer U predicts a raw similarity score for (q,d)
  - You can fine-tune U with contrastive examples: relevant and irrelevant

$$\mathbf{z} = \mathrm{BERT}(\mathrm{q}; [\mathrm{SEP}]; \mathrm{d})\,[\mathrm{CLS}]$$

$$\mathrm{score}(q,d) = \mathrm{softmax}(\mathbf{U}(\mathbf{z}))$$

Bi-encoder

- One BERT encoder to encode documents, do it just one time and store the document vectors
  - Again, you need to specify what is a “document”: a 100 words paragraph? Max input length of BERT? A sentence?
- The other BERT encoder encode the query, do it every time you have a query
- Then whenever we have a query, we just use encode the query and compute the dot product between it and every candidate document.

$$\mathbf{z}_q = \text{BERT}_Q(\text{q})\,[\text{CLS}]$$

$$\mathbf{z}_d = \text{BERT}_D(\text{d})\,[\text{CLS}]$$

$$\text{score}(q,d) = \mathbf{z}_q \cdot \mathbf{z}_d$$

ColBERT

A hybrid of TF-IDF and BERT contextual embeddings

- Instead of encoding the entire query or entire document into one embedding, we pre-store every word embedding in the document
- For each token in the query, find the most contextually similar token in d, then sum up these similarity scores

$$\text{score}(q,d) = \sum_{i=1}^{N} \max_{j=1}^{m} \mathbf{E}_{q_i} \cdot \mathbf{E}_{d_j}$$

Approximate nearest neighbor search

- Searching for the most relevant documents is essentially a problem of **nearest neighbor search**
- Modern systems use approximate nearest neighbor search algorithms like Faiss (Johnson et al., 2017)

Faiss

Faiss (Facebook AI Similarity Search) is a library that allows developers to quickly search for embeddings of multimedia documents that are similar to each other. It solves limitations of traditional query search engines that are optimized for hash-based searches, and provides more scalable similarity search functions.

Multi-hop architecture

- For some complex questions, we can use a more complex approach that involves multiple stages:
- Multi-hop architecture: A query is used to retrieve documents, which are then appended to the original query for a second stage of retrieval

Hallucination exists, even with RAG

Input Query

Are the deadlines established by the bankruptcy rules for objecting to discharge jurisdictional?

Westlaw AIAR Response

[A] **paragraph from the Federal Rules of Bankruptcy Procedure**, Rule 4007 states that the deadlines set by bankruptcy rules governing the filing of dischargeability complaints **are jurisdictional**. [...]

Input Query

What standard of review applies to abortion regulations under the U.S. Constitution?

Lexis+ AI Response

Currently, the "undue burden" test, as established in *Planned Parenthood of Southeastern Pa. v. Casey*, is the governing standard. Planned Parenthood of the Heartland, Inc. v. Reynolds, [...]. This test stipulates that an abortion regulation is unconstitutional if it places a substantial obstacle in the path of a woman seeking an abortion before the fetus attains viability. [...]

Input Query

Why did Justice Ginsburg dissent in Obergefell?

Practical Law Response

Justice Ginsburg dissented in Obergefell because she believed that the court's interpretation undermined Congress's intent under Section 601(a) (1) to protect copyright owners against unauthorized importation of low-priced, foreign-made copies of copyrighted works. She argued that the text of the Copyright Act clearly applies to works governed by the statute. [...]

Input Query

Do any special requirements apply to online dating services in Connecticut?

GPT-4 Response

Yes, Connecticut has specific legal requirements that apply to online dating services. Under Connecticut law, particularly **Conn. Gen. Stat. § 42-290**, online dating services that charge a fee and have members who reside in Connecticut must provide their users with specific safety awareness notifications. These notifications must include: [...]

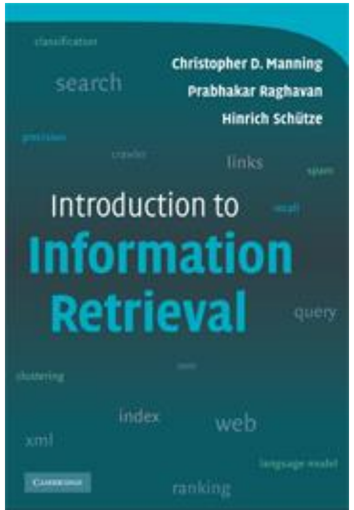
Hallucination-Free? Assessing the Reliability of Leading AI Legal Research Tools

Varun Magesh* Stanford University
Faiz Surani* Stanford University
Matthew Dahl Yale University
Mirac Suzgun Stanford University
Christopher D. Manning Stanford University
Daniel E. Ho† Stanford University

Pointers

- Relevant course: If you are interested in learning more about Information Retrieval, Northeastern offers **CS6200 / IS4200** which is a separate class dedicated to this topic!
- Textbook: Or you can read this textbook by Chris Manning, Prabhakar Raghavan, and Hinrich Schutze:

https://nlp.stanford.edu/IR-book/information-retrieval-book.html



Summarization

An introduction

Types of summaries

- **Extractive**:
  - Select key sentences / spans from DIRECTLY from the source text

“I was working. The work was on NLP. The TV was loud. My mom was watching the TV”

→ “I was working. My mom was watching the TV”

- **Abstractive / generative**:
  - Rewrites the content in new words, possibly introducing unseen phrases
  - More similar to how human would summarize

“I was working. The work was on NLP. The TV was loud. My mom was watching the TV”

→“Even though my mom was watching the TV and the volume was high,
I was working.”

Types of summaries

- Based on scope:
  - Single-document summarization:
    - One article, one paper
    - E.g. generate an abstract for an NLP research paper
  - Multi-document summarization
    - Multiple (related) texts
    - E.g. summarize the different approaches between a few papers, or summarize the overall contribution of a bunch of papers

Summarization can make IR more effective

- Document summaries or snippets can be used to indexing instead of the full text
- You can also first generate the summary (tl;dr) then use it for indexing
- Then retrieve using the summary embedding

Evaluation metrics

- BLEU
  - Usually for machine translation but can also be used here
  - N-gram based
  - Not as good as human eval of course,
  - Probably worse than other automatic eval metrics too
- ROUGE
  - Also n-gram based
  - Many variants
- Both need a reference summary

Evaluation metrics

- BERTScore
  - Uses embeddings
  - Capture meaning well
  - Computationall expensive, model-dependent
- MoverScore
- Sentence-BERT: can embed entire summaries using this model, then compute cosine similarity
- GPT or LLM based evaluation
  - “rate this summary from 1-5 for factual accuracy"